LINKÖPINGS UNIVERSITET
IDA
David Broman, Kristian Sandahl

Written exam for Software Engineering Theory

Date: 2008-08-22
Time: 14.00-18.00

Valid for pass of the written exam in courses: TDDB61, TDDB62, TDDC01, TDDC06, TDDC88, TDDC93.

Allowed aids: Two sheets of handwritten A4 pages. You may write on both pages, with any type of size and color. One volume of dictionary to/from English or an English wordbook.

Explicitly forbidden aids: Textbooks, machine-written pages, photocopied pages, pages of different format than A4, electronic equipment.

Graded exams will be handed out 2008-09-08 in Donald Knuth (IDA) between 12.30 and 14.00.

Questions of clarification will be answered by examiner David Broman (0707-909075), who will visit the exam about one hour after start.

Instruktioner till tentamensvakter

Studenterna får ha med sig 2 handskrivna A4-blad med text på båda sidorna och ett lexikon. Studenter med andra hjälpmedel utan särskilt tillstånd får inte påbörja tentamen förrän examinator kontaktats.

Instructions to students, please read carefully

- Try to solve as many problems as possible.
- Motivate all solutions.
- Please, write and draw clearly.
- Write only on one side of the paper.
- Write solutions for different areas (fundamental part) and different problems (advanced part) on separate sheets of paper.
- Label all papers with name and your Swedish personal number.
- You may write solutions in either Swedish or English.
- Please, note that the problems are not necessarily written in order of difficulty.

- **TIP!** Read through all exercises *in the beginning* of the exam and start with the ones you directly see a solution to. This will also give you the possibility to ask David questions about all parts of the exam, since he will visit you in the beginning of the exam time.

Grading

The exam consists of two parts: Fundamental and Advanced.

The Fundamental part has problems worth 10 credits per area. Areas are: Requirements, Design & Architecture, Testing, Planning & Processes, and Quality factors. Thus the Fundamental part can give maximally 50 credits.

The Advanced part has problems worth 50 credits in total. They can be distributed over two to five problems. Each problem typically requires a longer solution of several pages.

The maximum number of credits assigned to each problem is given in within parentheses at the end of the last paragraph of the problem.

Pass condition: At least 5 credits per area in the Fundamental part **and** at least 50 credits in total. This gives you the mark 3 in the Swedish system and an E in ECTS. Note that *no* credits are added, even if you passed the quiz exam in 2007.

Higher marks are given based on fulfilled *pass condition* **and** higher amounts of credits according to the following table:

Total credits	Mark in Swedish system
100-84	5
83-67	4
66-50	3
49-0	UK

Total credits	Mark in ECTS
100-96	A
95-83	B
82-68	C
67-55	D
54-50	E
49-0	F

Good Luck!

David & Kristian

Problems

Part 1: Fundamental

Area 1: Requirements

1a) Describe three different properties that written requirements in a natural language requirements specification shall have. (3)

1b) Explain the terms *non-functional requirement*, *design constraint* and *data dictionary.* (3)

1c) Describe 4 use-cases and draw a use-case diagram of a self scanning system in a shop. A self scanning system allows a user to scan European Article Number (EAN) codes of all articles, and deduct the sum from the customer's shopping account. Special solutions apply to alcohol, tobacco and non-packaged articles, such as, vegetables. (4)

Area 2: Design and Architecture

2a) Give at least two reasons for why it is important to design and document software architectures. Motivate your reasons by giving examples in a software project where a consulting company should deliver a booking system to a bus shuttle company. (3)

2b) Explain the differences and similarities between *coupling* and *cohesion* of modules, by giving examples from a standard software library (e.g. File I/O). In which case do we prefer high and/or low levels of these properties? (3)

2c) Draw a UML class diagram of your choice, which models a concrete system. The diagram should contain at least the following three types of relations between classes: *association*, *composition*, and *generalization*. State explicitly in the diagram the type of relation, i.e., print out if it is an association, composition, or generalization. Explain in detail the differences between these three relations by using your example diagram. (4)

Area 3: Testing

3a) State the main rationale for doing *equivalence class testing*, compared to *exhaustive testing*. (2)

3b) For each of the following statements, state if it is true or false. If you claim that a statement is false, give a rationale for why this is the case. (5)

1. System testing's main purpose is to validate the requirements.
2. When using a bottom-up integration strategy, it is especially important to create *stubs*.
3. JUnit is a testing framework for automated unit testing.
4. If a developer makes an error, it can lead to a fault in the software. If this software is executed, it can result in a failure.
5. Black-box testing is a method where the control-flow of the source code is statically examined.

3c) Imagine that you are the team leader of a software development team creating a client-server system for an online poker system. The system is built up of many modules (e.g. GUI module, encryption module, backup module, transaction module etc.) which should be integrated together in a reliable manner. Choose if you want to use a *bottom-up*, *top-down*, or a *big-bang* integration strategy. Explain the chosen strategy with an example and explain why you think that this approach is preferable over the other ones. State also if you see any drawbacks with the chosen strategy. (3)

Area 4: Planning and Processes

4a) Explain the main properties of a *project*. Discuss the differences between a project and a *process*. Give examples of both a typical project and a process. (3)

4b) Software Configuration Management (SCM) is critical when managing a successful software project.

- Give at least two reasons why the above statement is true. (2)
- Generally, SCM tools are necessary for efficient SCM. Name at least one open source or commercial SCM tool. Give a brief overview for how the tool is working by describing at least one of its principal functions. (2)

4c) Explain the main differences and similarities between the *waterfall model* and the *spiral model.* (3)

Area 5: Quality factors

5a) You have asked a technical novice customer to write down some initial requirements about the usability of a system the customer would order from you. You received the following:

1. The system shall fit well in our current routines.
2. The system shall be easy to learn by our users.
3. A user using the system shall process tasks much faster compared to manual routines.
4. Almost all users shall appreciate working with the system.
5. The visible interface shall not look too different compared to Windows applications.

Your task is now to write an answer to the customer where you give examples of how each of the requirements can be expressed more precisely. Skip all polite phrases and write the examples' part only. (5)

5b) Describe the terms *capability maturity level* and *(key) process area* in CMMI. (2)

5c) Explain the benefits for a software development organisation following three of the IS0 9001 principles:

- Continual improvement
- Factual approach to decision making
- Mutually beneficial supplier relationships

(3)

Part 2: Advanced

6) Imagine that you are the development manager for a software development company, which is creating geographic information system (GIS) products. A GIS product integrates hardware, software, and data for managing, capturing, analyzing, and displaying different geographic information, e.g., digital maps. The company is specialized on maps with sensitive data aimed for the military market. The system is based on a client-server architecture, where the server stores the map data, and the clients consist of various wireless handheld devices, which are used for analyzing and displaying maps. The company has 20 employees, where 11 are working in your development team.

a) You are currently using a development approach, which has similarities with the waterfall model. For project management, you are using a commercial tool. Typically, you use this tool for generating Gantt charts and calculating critical paths in the plan. Explain all phases in detail from idea to delivered product and draw the different parts in a Gantt chart. Note that your system is released in several versions and that the system is divided into several sub-systems during development. Explain how you create the critical path, locate the slack, and describe what use you have from them. (10)

b) You are also looking into switching to agile methods. Explain what this is and how it differs compared to you current approach. Give both pros and cons of switching. (5)

c) The methodology you have been looking into mostly is the Extreme programming (XP) approach. Explain in detail the 3 practices in XP that you find most appealing. (5)

d) 40% of the customer contracts have fixed price, i.e., the price and requirements are signed off in the beginning of a project. The rest of the contracts are based on payment per hour. Discuss pros and cons of your choice of software development model, depending on if the customer contracts have fixed price or if the payment is per hour. (5)

e) Select 3 quality factors that seem especially important for the company. Discuss how the choice of software lifecycle model can affect the quality factors. (5)

7) Imagine you are developing an auction system for private persons. The features include, but are not limited to:

- User account management
- Advertise design system
- Search for different articles to buy
- Auction management system
- Bidding system
- Fairly trusted payment system

Now you are supposed to describe either the entire system or a sub-system by:

a) Five functional requirements. (5)

b) A UML class diagram with at least five classes, where all classes have attributes and operations. (5)

c) A UML sequence diagram with at least 5 transactions and three roles (same as participants or objects). (5)

d) Five functional test cases with input and expected output. (5)